A Range-Based Localization Algorithm for Wireless Sensor Networks

Yuan Zhang, Wenwu Wu, and Yuehui Chen

Abstract: **Sensor localization has become an essential requirement for realistic applications over wireless sensor networks (WSN). In this paper we propose an ad hoc localization algorithm that is infrastructure-free, anchor-free, and computationally efficient with reduced communication. A novel combination of distance and direction estimation technique is introduced to detect and estimate ranges between neighbors. Using this information we construct unidirectional coordinate systems to avoid the reflection ambiguity. We then compute node positions using a transformation matrix $[T]$, which reduces the computational complexity of the localization algorithm while computing positions relative to the fixed coordinate system. Simulation results have shown that at a node degree of 9 we get 90% localization with 20% average localization error without using any error refining schemes.**

Index Terms: **Localization algorithm, range-based, wireless sensor networks.**

I. INTRODUCTION

Unlike other networks with more logical structures, wireless sensor networks (WSN) are closely related to the geometric environment where they are deployed. In particular, location information has been proven to be very useful in the design of sensor network infrastructures. First, location information is needed to identify the location where an event originates. Second, location information can assist in a lot of system functionalities such as geographical routing, network coverage checking [1], and location based information querying [2]. Third, location awareness can facilitate application services such as location directory service that provides doctors with the information of nearby medical equipments and personnel in a smart hospital.

Due to the energy constraint and the limited processing capability of the tiny sensor nodes, however, the localization technique must be designed such that it is energy-efficient, does not require much information, and is computationally simple. Moreover, due to the ad hoc nature of the WSN, the localization technique must not depend on any infrastructure requirements. Finally, the localization technique must be robust with good resolution to ensure accurate and precise localization.

A number of localization techniques have been described in literature, most of which either rely on some sort of an infrastructure, or, they use a fraction of anchor (position-aware) nodes, to induce location-awareness in sensor nodes. Both these classes of techniques are not suitable for the WSN applications we refer to here simply because of the inability to set up any infrastructure or provide anchor nodes in the environments that we address. The truly ad hoc localization techniques that exist, and are discussed in literature involve a lot of communication overhead, and/or involve a great deal of computational complexity which renders the localization technique itself as an energy-inefficient component of the network protocols.

In this paper, we present an ad hoc localization technique which satisfies the requirements mentioned earlier. Our first contribution lies in using a combination of distance (time difference of arrival) and direction (angle of arrival) estimation techniques to detect and estimate ranges between neighbors and their directions. Based on this information we can construct unidirectional coordinate systems to avoid the reflection ambiguity. The key concept is that one can eliminate the reflection ambiguity by propagating information between sensors of only 1-hop neighborhood (rather than 2-hop neighborhood as are proposed by other papers). This reduces the amount of information, and thus communication, required to induce location-awareness in sensor nodes. Our second contribution lies in developing a transformation matrix, $[T]$, at each node to convert the positions of the neighbors from node coordinate system to a fixed coordinate system while reducing the computational complexity. These contributions thus help in reducing the computational and communication complexity.

The rest of the paper is organized as follows. We give a background and discuss some of the existing work done in the field of localization in Section II. In Section III, we discuss our localization technique in detail. In Section IV, we discuss the error sources. We present and discuss our simulation results in Section V. Finally, we conclude our work in Section VI.

II. BACKGROUND ON LOCALIZATION

A. General Background

Many localization systems for mobile computing applications exist today. Some examples are GPS, Active Badge [3], Active Bat, Cricket [4], RADAR, and SpotON. A survey and taxonomy of these systems is provided in [5]. All the systems discussed in that survey require an infrastructure set-up and are geared more towards location sensing for office applications than for ad hoc WSN. Hence, we cannot use them in truly ad hoc WSNs. In general, all the current localization systems for ad hoc sensor networks can be viewed as a 2-step process as explained below.

A.1 Distance or Range Estimation

This involves finding distances between the nodes using one of the several existing ranging techniques.

Manuscript received June 29, 2005; approved for publication by Kung Yao, Guest Editor, November 30, 2005.

Y. Zhang and Y. Chen are with the School of Information Science and Engineering, Jinan University, Jiwei Road 106, Jinan 250022, Shandong, P. R. China, email: {yzhang, yhchen}@ujn.edu.cn.

W. Wu is with the Network Information Center, Jinan University, Jiwei Road 106, Jinan 250022, Shandong, P. R. China, email: wuww@ujn.edu.cn.

(1) Timing-based (TOA, TDOA): The distance between the two nodes is estimated by timing the flight of a communication signal.
(2) Directionality-based (AOA): Instead of estimating the distance, the direction of the received signal is estimated to infer the angle of the sender node.
(3) Signal-strength-based (RSSI): The power of a received signal is used to estimate the distance between two nodes based on theoretical or empirical models.
(4) Hop-based (DV-HOP, Hop-TERRAIN): Position-aware anchors flood their locations. The nodes maintain a minimum hop count to different anchors. The distance to these anchors is estimated by multiplying the hop count with the average distance per hop.

A.2 Position or Location Computation

This step involves calculating the position of a node relative to a fixed coordinate system using a mathematical technique such as Lateration, Multilateration, Triangulation, etc.

One can also apply a third step, refinement, to improve upon the position estimates obtained from the above 2-step process. Whether or not refinement improves the accuracy of a localization system depends upon the accuracy of initial position estimates which in turn depends upon the accuracy of range estimation.

B. Classification Scheme for Localization Systems

We classify the current localization systems as range-based and range-free systems, based on the techniques they use for distance estimation and position computation. Other forms of classification schemes exist. However, they are out of the scope of this paper.

Range-based systems, or fine-grained localization systems, utilize timing-based, directionality-based, or signal-strength-based techniques for distance estimation, and the position of the node is computed using multilateration or triangulation. Systems that fall into this category are [6]–[11]. As distance estimates are fairly accurate, refinement may not yield a great deal of improvement in the initial position estimates as shown in [12].

Range-free systems, or coarse-grained localization systems, utilize hop-based techniques for distance estimation that infer the proximity of a node to some reference points and use centroid calculations or area-based triangulation to estimate the nodes positions. Systems that fall into this category are [13]–[17]. Refinement may greatly improve the initial position estimates as the distance estimation may be more erroneous as compared to range-based systems. Some systems, [18], can fall into either category since they do not rely on using a particular technique for distance estimation. Instead, they give the option to use any technique. Our proposed localization algorithm falls into the category of range-based systems.

C. Review of Current Localization Systems

Of the many existing localization systems for WSN, the ones discussed in [6]–[8] and [13] require some sort of an infrastructure setup (mounting reference points or beacons at known locations on walls, ceilings, etc.) for the localization system to work. Providing such an infrastructure is not possible in ad hoc networks that our research addresses thus we discard the use of these localization techniques.

Other localization systems developed in [9], [10], and [14]–[16] use heterogeneous nodes, i.e., a mix of location-aware anchor nodes and normal sensor nodes. In these solutions some fraction nodes have the ability to determine their locations, and the other nodes are simple nodes. The locationing accuracy of such systems heavily depends on the number and distribution of the anchor nodes, which in turn is susceptible to uncertainties of random node distribution upon deployment. Also, in some environments it is difficult to localize the anchors themselves. Further, for a simple node to be able to localize itself it must be connected to at least 3 anchors. These factors impose constraints on the localization system and thus we avoid the use of these techniques.

The localization systems SPA [11] and AFL [18] support truly ad hoc localization and their infrastructure-free and anchor-free operation makes them suitable for use in wireless ad hoc networks. Our work is perhaps most closely related to the SPA algorithm. In the SPA algorithm, the nodes first find their neighbors and the distances to them by using a timing-based ranging technique. They exchange this information with their neighbors and use it to construct a local coordinate system. Then they build a network coordinate system based on the density factor of an n-hop neighborhood. Finally, the algorithm determines the positions of the nodes. Our algorithm, as discussed in Section III, is computationally simpler, and reduces the amount of information exchanged during localization as compared to the SPA algorithm.

III. PROPOSED LOCALIZATION ALGORITHM

The proposed localization algorithm follows the 2-step process of range estimation and position computation discussed in Section II-A. In the following sub-sections we discuss our algorithm in detail.

A. Node Initialization

Nodes are first deployed in the environment to be monitored by sprinkling them using unmanned air vehicles (UAVs) flying over the environment. The nodes land at random positions within the environment boundary. Now, every node x starts exchanging beacons to detect its 1-hop neighbors y. We denote this set of neighbors by H_x. Two nodes, x and y, are considered 1-hop neighbors if they can communicate with each other directly.

For $\forall y \in H_x$, every node x uses a timing-based ranging technique, such as time-of-arrival (TOA) or time-difference-of-arrival (TDOA), to measure the time of a one-way or a roundtrip flight of a communication signal. By adopting an appropriate model for the propagation speed of the communication signal the node can estimate the distance to each of its neighbors as in Fig. 1. We denote this set of distances, for any node x, by d_{xy}.

Further, every node x uses directionality-based angle-of-arrival (AOA) method to estimate the direction of each of the 1-hop neighbors. By using the hardware geometry and the range information of 1-hop neighbors, every node x computes the approximate angles at which the 1-hop neighbors lie. We denote

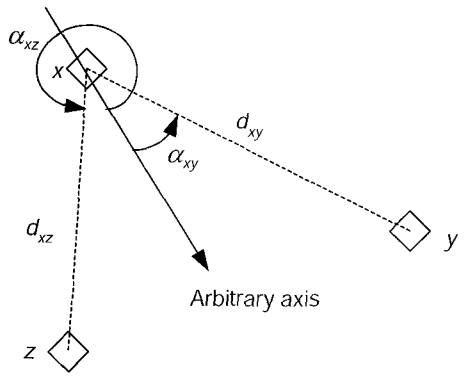


Fig. 1. Distance and direction estimation.

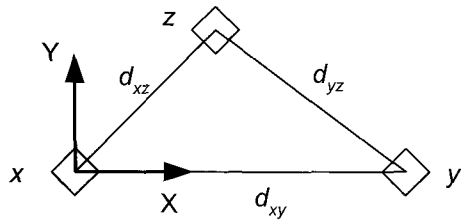


Fig. 2. Node coordinate system case 1.

this set of angles, for any node x, by α_{xy}. Each node has an arbitrary axis against which all the angles are measured, and for consistency, all angles are measured in the same sense as shown in Fig. 1. While the AOA method is not very accurate we can still rely on it as we do not need accurate angle estimates, instead, we only need to infer the directions in which the 1-hop neighbors lie.

Finally, all the 1-hop neighbors exchange their H_x, and d_{xy} data with each other. Such an exchange is required for the nodes to have enough information to compute positions of the neighbors. As a result of node initialization every node x knows its 1-hop neighbors, 2-hop neighbors, distance to 1-hop neighbors, distances between 1-hop and 2-hop neighbors, and angles at which 1-hop neighbors lie.

B. Node Coordinate System (NCS)

One key step of localization is to determine the physical coordinates of a group of sensor nodes, which form an NCS. Here the coordinates are relative, meaning that they present an arbitrary rigid transformation (rotation, reflection, and translation) system. As recognized in the SPA algorithm [13], for a node x to build its NCS, NCS_x, it must have 2 neighbors y and z such that they are all non-collinear 1-hop neighbors of each other, i.e.,

(1) x, y, and z should be non-collinear, and,
(2) the distances d_{xy}, d_{xz}, and $d_{yz} > 0$ should be known at node x.

This information can be easily derived for any node x given the data gathered from node initialization. Thus, the NCS_x can be defined such that node y lies on the +X-axis and node z lies in the direction of +Y-axis with the origin of the coordinate system located at the node x itself.

Now, consider the two scenarios in Figs. 2 and 3. Following the NCS construction process stated above, for identical topology there exist two orientations of NCS_x based on which neighbor node x selects for its +X-axis. Since the nodes lack the control over creating unidirectional coordinate systems, this results in a reflection ambiguity later on when we compute the node positions relative to a fixed coordinate system. Resolving this ambiguity at that time requires information about the topology of 1-hop neighbors, which induces a communication and a computation overhead. Moreover such information may not be available.

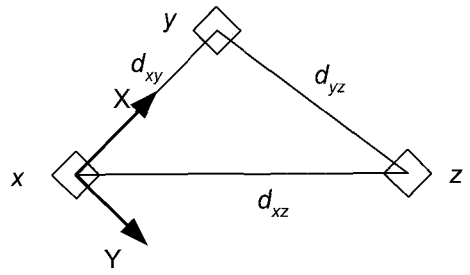


Fig. 3. Node coordinate system case 2.

To avoid such ambiguity the selection must be based on some decision-making process so that unidirectional NCSs are constructed. We thus require that node must also know the directions (i.e., α_{xy} and α_{xz} as shown in Fig. 1) in which y and z lie. Node x has this angle information as a result of directionality-based ranging during node initialization. The nodes can thus apply the following procedure to select the appropriate neighbor for +X-axis, which in turn will ensure the creation of unidirectional NCSs at every node.

if$(|\alpha_{xy} - \alpha_{xz}| > 180)$
    +X-*axis* = *node with MAX*(α)
else
    +X-*axis* = *node with MIN*(α).

This procedure guarantees that NCS at any node is always constructed such that +Y-axis is 90 degree counter-clockwise to +X-axis when viewed from the top. The reflection ambiguity while computing the node positions relative to a fixed coordinate system is therefore avoided.

C. Local Position Computation

Now, every node x can compute the positions of its 1-hop neighbors relative to its NCS. This is the local position of the neighbor, and is denoted as ${}^xP_y$. It is defined by the x-coordinate, the y-coordinate, and the angle at which the neighbor lies relative to the +X-axis of NCS_x, i.e., ${}^xP_y = (y_{\mathrm{X}}, y_{\mathrm{Y}})$.

As shown in Fig. 4, for any node x, local position computation starts with the nodes y and z which are the nodes used to construct NCS_x . Hence the position of node y w.r.t. x, ${}^xP_y$, simply becomes

$$y_{\mathrm{X}} = d_{xy},\ y_{\mathrm{Y}} = 0 \tag{1}$$

and the position of node z w.r.t. x, ${}^xP_z$, becomes

$$z_{\mathrm{X}} = d_{xz}\cos\theta_z,\ z_{\mathrm{Y}} = d_{xz}\sin\theta_z \tag{2}$$

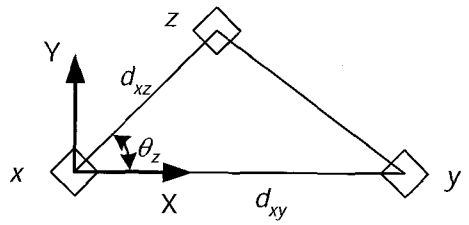


Fig. 4. Local position computation initiation.

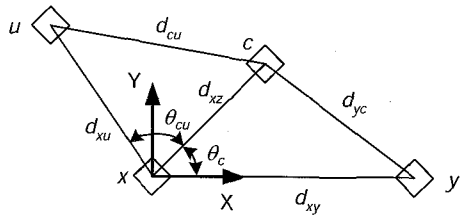


Fig. 5. Local position computation.

where θ_z can be obtained by using the cosine rule as

$$\theta_z = \arccos \frac{d_{xz}^2 + d_{xy}^2 - d_{yz}^2}{2 d_{xz} d_{xy}}. \tag{3}$$

As shown in Fig. 5, the position of any node u, ${}^xP_u$, can be computed if it is the neighbor of a node c such that the position of c is known relative to the NCS_x, and the distances d_{xu} and d_{cu} are known. Thus ${}^xP_u$ can be computed as

$$u_{\mathrm{X}} = d_{xu} \cos\theta_u, \; u_{\mathrm{Y}} = d_{xu} \sin\theta_u \tag{4}$$

where

$$\theta_u = \theta_c \pm \theta_{cu}. \tag{5}$$

In (5), θ_c is known from the position of c, and θ_{cu} is obtained by applying the cosine rule to the triangle cxu. Since only the value of θ_{cu} can be determined by the cosine rule and not its direction there exists a $\pm$ ambiguity. We use the direction information, α, of the nodes u and c to resolve the $\pm$ ambiguity as follows.

if$(((|\alpha_{xu} - \alpha_{xc}| > 180)$ *AND* $(\alpha_{xu} < \alpha_{xc}))$ *OR*
$((|\alpha_{xu} - \alpha_{xc}| < 180)$ *AND* $(\alpha_{xu} > \alpha_{xc})))$
$\theta_u = \theta_c + \theta_{cu}$
else
$\theta_u = \theta_c - \theta_{cu}$.

A comparison of the proposed algorithm with the algorithm presented in [13] reveals that, for local position computation, the proposed technique requires a position-unaware node to be a neighbor of just one position-aware node as opposed to that of [13], which requires a position-unaware node to be the neighbor of at least two position-aware nodes. In latter, they use the second neighbor to resolve the ambiguity to determine the true position of the position-unaware node from the two possibilities. Since a second neighbor may not exist, the information relative to it may be difficult to obtain. As a result, the node may be unable to compute the local position of its neighbors. However, as our algorithm does not require the information relative to the second neighbor, it maximizes the 1-hop neighbors for which the local positions can be computed. Also, the computation is reduced and simplified as the amount of information handled is reduced.

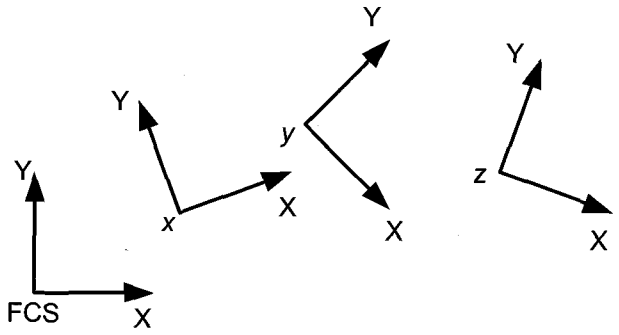


Fig. 6. Network position computation.

After local position computation, each node x sends the neighbors position information to its respective neighbors. Thus the neighbors of x know their positions relative to NCS_x. As this happens at all the nodes, node x also knows its positions relative to the NCS of its neighbors. While this is required we recognize that such an exchange is not required at this stage. It can be delayed till the node develops its transformation matrix, as shown in Section III-D.1. Moreover, the nodes need not be informed of its local position relative to all its neighbors unless it is going to use all that information. In our algorithm, the node only needs this information from the node relative to which it constructs its transformation matrix. These simple strategies can be used to reduce the communication overhead resulting from exchanging redundant or non-useful data.

D. Network Position Computation

In this part we compute the position of a node relative to a fixed coordinate system (FCS). First we develop a transformation matrix $[T]$ at every node, which converts the local position (relative to an NCS) to a network position (relative to the FCS) using the equation

$${}^{FCS}P = {}^{FCS}[T]_{NCS} \cdot {}^{NCS}P \tag{6}$$

where
${}^{FCS}P$ =network position relative to the FCS
${}^{NCS}P = {}^xP_y$= local position relative to the NCS, and
${}^{FCS}[T]_{NCS}$ = Transformation matrix.

We illustrate network position computation by considering the example in Fig. 6. We defer the discussion of FCS until Section III-D.2. For now we assume that there exists an FCS such that the nodes around it know their positions relative to it. Also these nodes know the position of the FCS relative to their NCSs. Given this, any node x can compute ${}^{FCS}[T]_x$. Since it knows ${}^xP_y$ it can find the position of its neighbors y relative to the FCS, ${}^{FCS}P_y$, as

$${}^{FCS}P_y = {}^{FCS}[T]_x \cdot {}^xP_y. \tag{7}$$

Now x can send ${}^{FCS}[T]_x$, ${}^{FCS}P_y$, and ${}^xP_y$ to all y. Note that while ${}^xP_y$ was computed earlier in the cycle, it is transmitted only now as x has developed its transformation matrix. We

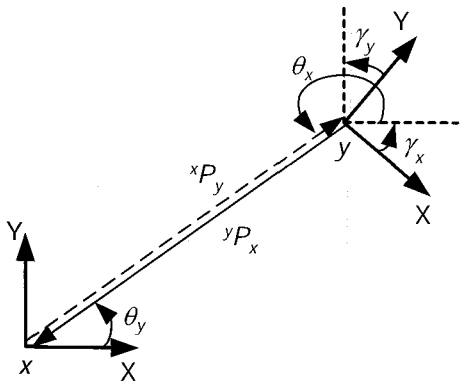


Fig. 7. Transformation matrix development.

discussed this transmission delay in Section III-D. Now, y can compute ${}^x[T]_y$, and thus y can obtain ${}^{FCS}[T]_y$ as

$$ {}^{FCS}[T]_y = {}^{FCS}[T]_x \cdot {}^x[T]_y. \tag{8} $$

Since y knows ${}^yP_z$ it can find the position of its neighbors z relative to the FCS, ${}^{FCS}P_z$, as

$$ {}^{FCS}P_z = {}^{FCS}[T]_y \cdot {}^yP_z. \tag{9} $$

Now y can send ${}^{FCS}[T]_y$, ${}^{FCS}P_z$, and ${}^yP_z$ to all z. Thus, z can compute ${}^y[T]_z$, and z can obtain ${}^{FCS}[T]_z$ similar to y, and it can find the position of its neighbors relative to the FCS.

This process continues until all the nodes know their positions relative to the FCS. At this point, the nodes in the network that were unable to build their NCS and/or were unaware of their local positions relative to any neighbor, can compute their positions relative to the FCS if they have three neighbors whose position are computed relative to the FCS. They can use triangulation to localize themselves.

We summarize that in order to proceed with network position computation we must have the following

(1) fixed coordinate system (FCS) definition and
(2) transformation matrix $[T]$ at any node.

D.1 Transformation Matrix Development

Before getting into the details of how to define an FCS, let us see how to find the transformation matrix for any given node. The problem we are trying to solve is

Given two coordinate systems, defined relative to each other, find a transformation matrix $[T]$ that converts the position relative to one coordinate system to a position relative to the other coordinate system.

Consider the scenario in Fig. 7. Two nodes, x and y, are 1-hop neighbors of each other, and their NCSs have been constructed. ${}^xP_y$ and ${}^yP_x$ are known at the respective nodes as a result of local position computation. We assume that node x has developed ${}^{FCS}[T]_x$ since it lies in the vicinity of the FCS. It also computes ${}^{FCS}P_y$. Let us develop the transformation matrix ${}^x[T]_y$ at node y such that it converts a local position to a position relative to x, and thus to a position relative to the FCS. To do so, y needs to know ${}^xP_y$, so x must transmit ${}^xP_y$ to y. As we saw in Section III-D, the nodes exchange local positions only if required. x also transmits ${}^{FCS}[T]_x$ and ${}^{FCS}P_y$ but they are not needed for developing ${}^x[T]_y$. Now y can find the orientation and the position of NCS_y relative to NCS_x, based on which ${}^x[T]_y$ is given by

$$ {}^x[T]_y = \begin{bmatrix} {}^x[R]_y & {}^xP_y \\ 0\ 0 & 1 \end{bmatrix} \tag{10} $$

where

${}^x[R]_y$ = orientation of NCS_y relative to NCS_x and
${}^xP_y$ = position of y relative to NCS_x.

In (10), ${}^xP_y$ is known, and ${}^x[R]_y$ is given by

$$ {}^x[R]_y = \begin{bmatrix} t_{11} & t_{12} \\ t_{21} & t_{22} \end{bmatrix}. \tag{11} $$

The columns of ${}^x[R]_y$ are the direction cosines of the unit vectors along the X and Y-axis of NCS_y relative to NCS_x. In order to compute these elements, we must know the angles γ_x and γ_y as shown in Fig. 7. Since we have ensured the construction of unidirectional coordinate systems we can write

$$ \gamma_x = \gamma_y = \gamma \tag{12} $$

where γ is the angle by which NCS_y must be rotated to be aligned with NCS_x. It is given by

$$ \gamma = \pi + \theta_x - \theta_y. \tag{13} $$

Substituting (11) and (12) into (10) gives the transformation matrix, ${}^x[T]_y$, as

$$ {}^x[T]_y = \begin{bmatrix} \cos\gamma & \sin\gamma & y_X \\ -\sin\gamma & \cos\gamma & y_Y \\ 0 & 0 & 1 \end{bmatrix}. \tag{14} $$

Further, using (13), (14) can be written as

$$ {}^x[T]_y = \begin{bmatrix} \cos(\pi + \theta_x - \theta_y) & \sin(\pi + \theta_x - \theta_y) & y_X \\ -\sin(\pi + \theta_x - \theta_y) & \cos(\pi + \theta_x - \theta_y) & y_Y \\ 0 & 0 & 1 \end{bmatrix} \tag{15} $$

where y_X, y_Y, θ_x, and θ_y are all known from ${}^xP_y$ and ${}^yP_x$.

Based on (15), the transformation matrix ${}^x[T]_y$ can be computed for any node y, which in turn can be used to convert a position relative to y to a position relative to x and thus a position relative to the FCS.

One might have expected that as a multihop network gets larger the nodes farther away from the data-sink (see below) might suffer from error propagation due to angle measurement errors in the several transformation matrices. It thereby may increase the localization error. As demonstrated in [19], however, for both distance measurements and angle measurements the localization error is not greatly affected as the network scales.

D.2 Fixed Coordinate Systems (FCS)

In this section, we see how to define the fixed coordinate system (FCS) which acts as a reference relative to which all the nodes in the network are localized. To avoid the overhead of updating the location information due to the change in reference,

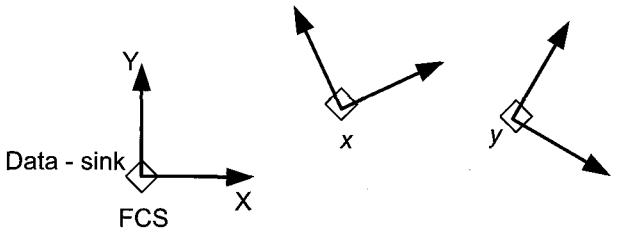


Fig. 8. Fixed coordinate system.

the FCS must be defined such that it does not change its direction and orientation too often. One such coordinate system is defined in [11] where they use the density factor of an n-hop neighborhood to define a coordinate system as a function of the position of the nodes in the n-hop neighborhood. While this approach is usable, there is a lot of communication overhead involved in broadcasting the location information over the n-hop neighborhood.

Since the nodes in an ad hoc network usually monitor the activities in a remote, and/or hostile environment, and gather and report the information about such an environment to a fixed data-sink. The data sink then transmits the data to the end user over the Internet or by some other means of communication. Due to the presence of such a fixed data-sink, it makes sense to exploit it by using it for defining the FCS.

Consider the scenario shown in Fig. 8. In Fig. 8, the data-sink is treated as a normal node and it runs the node initialization process to derive the information about the nodes in its vicinity. By using this information it constructs its NCS, which is the FCS, based on the procedure discussed in Section III-B. Henceforth, we will call this data-sink NCS as FCS. Now the data-sink computes the positions of the nodes in its vicinity relative to its FCS and transmits this information. Thus the nodes in the vicinity of the data-sink become aware of their positions relative to the FCS. Since this entire process is simultaneously happening at all the other nodes, the nodes in the vicinity of the data-sink will also know the position of the data-sink relative to their NCSs. Now these nodes can find the ${}^{FCS}[T]_x$ matrix, and begin network position computation to induce location-awareness in the nodes relative to a fixed coordinate system.

IV. DISCUSSION OF ERRORS

A. Error Sources and Assessment

Our localization algorithm is a range-based technique which uses a novel combination of distance and direction estimates, i.e., range estimates, to estimate positions. The accuracy of localization thus directly depends on the accuracy of range estimates. Due to the irregularities arising from the noise present in the environment, such as multi-path interference, signal fading, obstructions blocking line-of-sight, signal reflections, etc., the radio propagation model is not ideal, and the range estimates are not the most accurate. As a result, the computed position deviates from the actual position of the node, thus inducing error in the position estimates. We denote this error as the localization error, and we can define it as

$$LE = \sqrt{(\mathrm{X}_e - \mathrm{X}_a)^2 + (\mathrm{Y}_e - \mathrm{Y}_a)^2} \tag{16}$$

where $(\mathrm{X}_e, \mathrm{Y}_e)$ are the estimated coordinates, and $(\mathrm{X}_a, \mathrm{Y}_a)$ are the actual coordinates of a node.

Since the position of a node may be estimated by n neighboring nodes, we can assess the localization error for any node as

$$LE = \sqrt{(\frac{\sum \mathrm{X}_e}{n} - \mathrm{X}_a)^2 + (\frac{\sum \mathrm{Y}_e}{n} - \mathrm{Y}_a)^2} \tag{17}$$

where $(\mathrm{X}_e, \mathrm{Y}_e)$ are the estimated coordinates of a node by each of its n neighbors.

Finally, we can assess the mean localization error of the localization algorithm as

$$LE_{mean} = \frac{1}{N} \sum LE. \tag{18}$$

For assessing the mean localization error of our localization algorithm we only consider the nodes whose position relative to the FCS has been computed.

B. Error Control and Position Refinement

To reduce the localization error and improve the accuracy of localization we may use appropriate error control and position refinement algorithms. However, despite the error sources discussed in the previous section, range estimates obtained by a range-based system are relatively fairly accurate, and thus the position estimates are also fairly accurate. So, as pointed out in [12] the error control and refinement algorithms do not improve the accuracy of a range-based localization system significantly. Nonetheless, due to error propagation, the localization error may increase with distance, and small errors may be amplified and cumulated into large errors. We briefly discuss various error control methods and refinement algorithms that can be used to reduce the effects of error propagation and thus reduce localization error, and improve the accuracy of localization.

To improve direction estimation and reduce the effect of error propagation thereby, [9] suggests some simple techniques that can be employed to reduce the error by about half. One algorithm is based on the fact that small angles are more error prone than large angles. Thus, decision-making based on small angles must be avoided. They recommend the use of threshold values to eliminate small angles. Another algorithm is to eliminate the outliers when multiple position estimates are obtained. This is a refinement technique and is done by iterative centroid computation and outlier elimination after each computation till an acceptable position estimate is reached.

Another simple and popular refinement algorithm to mitigate error propagation is suggested in [15] where they use a confidence value with the position estimates to weigh the equations of an over-determined system. Then, they solve this over-determined system of equations by a least square solver to average out the errors.

The choice of an error control method or a refinement algorithm, and the decision whether to use it or not may be governed by the accuracy of localization desired, which in turn may

depend upon the application that uses the location information such as routing, data querying, etc. Nonetheless, the algorithm adopted to reduce the error must be efficient and should not involve high computational complexity to ensure that localization is not rendered as a resource-consuming component of the network protocol. In our simulations, we do not use any error refinement algorithm; however, if deemed necessary we can implement one.

V. SIMULATION

A. Simulation Scenario and Performance Metrics

We simulate our localization algorithm by deploying n sensor nodes in a square region R such that the node positions are generated using a random uniform distribution. We select the values of n and R such that with a communication range r of 10 m we obtain a node degree varying from 5 to 15. For example, to obtain $d = 8$ with $r = 10$ m in a region $R = 200$ m $\times$ 200 m we deploy $n = 1000$ sensors. We define node degree as the expected number of neighbors that any given node will have after deployment. We also assume that the range r is perfectly circular and that it is not only the communication range but also the ranging distance i.e., the range up to which the distance can be estimated. Further, we assume that the ranging error is Gaussian with a fixed standard deviation, for both distance and direction estimation. We assume a ranging error of 1% for all our simulations. Most of these assumptions are consistent with those in the literature.

Our aim here is not to study the performance of an error refinement algorithm but to investigate whether we can improve the localization extent with a lower node degree. We evaluate the performance of our localization algorithm based on 2 performance metrics (localization extent, and localization error) explained in the following sections, which are averaged over 100 simulation runs.

A.1 Localization Extent

We define localization extent as the number of nodes for which the position is determined relative to the FCS. We express the localization extent as a fraction of deployed nodes that are able to localize themselves relative to the FCS at a given node degree. For example, 90% localization extent would mean that 90% of the deployed nodes were able to estimate their positions relative to the FCS.

A.2 Localization Error

As defined in Section IV-A localization error is the deviation of the estimated node location from the actual node location. We express this metric relative to the communication range, in terms of percentage. For example, 5% localization error would mean that the deviation of the estimated position from the actual position is 0.05 r.

B. Numerical Results and Discussion

We ran 100 simulations of the scenario discussed in Section V-A for each d varying from 5 to 15 to obtain the values of the performance metrics. Then the results are averaged and plotted to evaluate the performance of our localization algorithm.

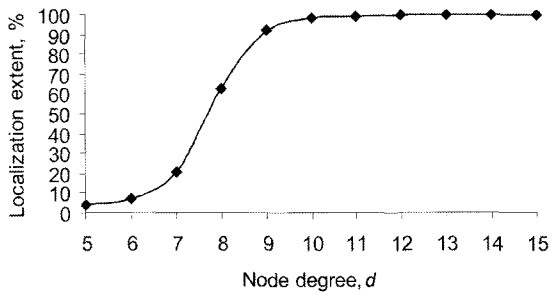


Fig. 9. Localization extent vs. node degree.

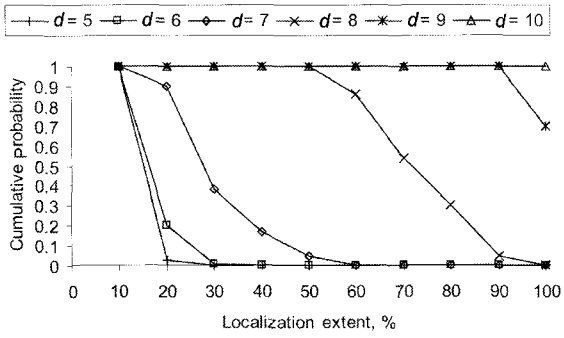


Fig. 10. Probability distribution of localization extent.

Fig. 9 shows a graph of localization extent vs. node degree. We observe that with $d \leq 7$ the localization extent is very low. This is due to the inability of the nodes to compute the local position of all the neighbors that it may have detected. As a result, a node may not be able to develop a $[T]$ matrix that converts the local positions to positions relative to the FCS. This may result in a low localization extent.

However this problem is resolved as the node degree increases, making the deployment denser. As seen in Fig. 9 at $d = 9$, an average localization extent of 90% or more is obtained and beyond this 100% localization is obtained almost all the time.

The probability of obtaining a certain localization extent at a given value of d is shown in Fig. 10. We only plot these curves for values of d varying from 5 to 10 because for $d = 11$ or more we get 100% localization all the time. The graph shows that at $d = 9$ the probability of obtaining a localization extent of 90% is 0.9. Further, at $d = 10$ we get 100% localization each time as indicated by the plot. This is also true for $d = 11$ and more.

We illustrate average localization error for different values of d in Fig. 11. At $d = 8$ or less the localization error is less than 10% of the communication range r. This high accuracy of localization can be mainly attributed to a low localization extent at those values of d due to which the increase in error due to propagation is fairly small. However as the value of d is increased to 9 or more the localization extent is increased, as seen in Fig. 9, as a result of which the error is propagated further in the network

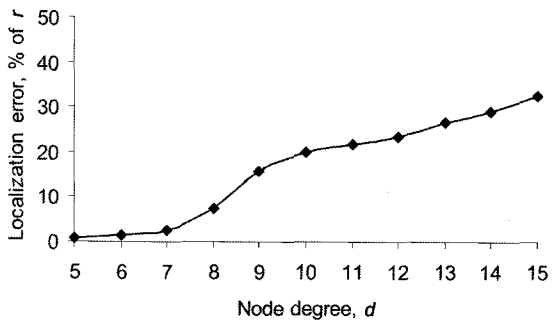


Fig. 11. Localization error vs. node degree.

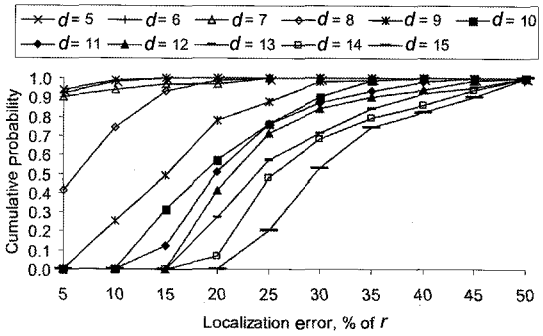


Fig. 12. Probability distribution of localization error.

and the average value of localization error increases. At $d = 9$ the average localization error is 20% of r, i.e., an error of 2 m for $r = 10$ m. If the application demands a higher level of localization accuracy, by using the simplest of error refining schemes, such as centroid computation, the error can be reduced.

Finally, in Fig. 12, we plot the probability of the average localization error being less than a certain value for different values of d. The graph shows that as the value of d increases the probability of the average localization error being less than 10% of r is greatly reduced. For $d = 5, 6$, or 7 the localization error is less than 5% with a probability of 0.9 or more. However, at $d = 9$ we can only guarantee with a probability of 0.8 that the average localization error will be less than 20%. Again, this can be improved by using simple error refining schemes that improve the position estimates and reduce the localization error.

VI. CONCLUSION AND FUTURE WORKS

In this paper, we have presented an infrastructure-free, anchor-free, and computationally simple ad hoc localization algorithm for wireless sensor networks. Our first contribution lies in using a novel combination of distance and direction estimation techniques to estimate the positions of nodes in an ad hoc network without giving rise to reflection ambiguities. As a result, the information exchange required to induce location-awareness in the sensor nodes is reduced requiring the nodes to communicate only while detecting neighbors and while exchanging the final position information.

Our second contribution lies in developing a transformation matrix $[T]$ at each node to convert the local positions of the neighbors to positions relative to a fixed coordinate system. This contribution demonstrates the application of a simple computation method to the localization problem for reducing the computational complexity. Further, the simulation results obtained by averaging 100 runs of our localization algorithm have shown that at a node degree of 9 we get 90% localization with 20% average localization error without using any error refining schemes.

Potential issues exist to refine upon the discussion of this localization algorithm. One of the main concerns is to perform quantitative comparisons to other range-based localization systems. For instance, numeric illustration through simulation can undisputedly demonstrate the reduction in information exchange compared to previous algorithm, thus showing the performance efficiency.

Extrinsic sensor measurement error is attributed to the physical effects on the measurement channel, such as the presence of obstacles, multipath and shadowing effects and changes in the signal propagation speed due to changes in the surrounding environment. Such non-standard error scenario is also worthy of study to further check the robustness of the algorithm.

ACKNOWLEDGMENTS

This work is supported by NSFC under Grant No. 60573065 and the Science & Technology Foundation of Jinan University under Grant No. Y0519 and Y0520.

REFERENCES

[1] S. Meguerdichian, F. Koushanfar, M. Potkonjak, and M. B. Srivastava, "Coverage problems in wireless ad-hoc sensor networks," in *Proc. IEEE INFOCOM 2001*, (Ankorange, USA), Apr. 2001, pp. 1380–1387.

[2] H. Gupta, S. R. Das, and Q. Gu, "Connected sensor cover: Self-organization of sensor networks for efficient query execution," in *Proc. MobiHoc 2003*, (Annapolis, USA), June 2003, pp. 189–200.

[3] R. Want, A. Hopper, V. Falcao, and J. Gibbons, "The active badge location system," *ACM Trans. Information Systems*, vol. 10, pp. 91–102, Jan. 1992.

[4] N. B. Priyantha, A. Chakraborty, and H. Balakrishnan, "The cricket location-support system," in *Proc. ACM MobiCom 2000*, (Boston, USA), Aug. 2000, pp. 32–43.

[5] J. Hightower and G. Borriello, "Location systems for ubiquitous computing," *Computer*, vol. 34, pp. 57–66, Aug. 2001.

[6] A. Ward, A. Jones, and A. Hopper, "A new location technique for the active office," *IEEE Pers. Commun.*, vol. 4, pp. 42–47, Oct. 1997.

[7] L. Girod, V. Bychkovskiy, J. Elson, and D. Estrin, "Locating tiny sensors in time and space: A case study," in *Proc. ICCD 2002*, (Freiburg, Germany), Sept. 2002, pp. 195–204.

[8] P. Bergamo and G. Mazzini, "Localization in sensor networks with fading and mobility," in *Proc. IEEE PIMRC 2002*, (Lisbon, Portugal), Sept. 2002, pp. 750–754.

[9] D. Niculescu and B. Nath, "Ad hoc positioning system (APS) Using AOA," in *Proc. IEEE INFOCOM 2003*, (San Francisco, USA), Apr. 2003, pp. 1734–1743.

[10] A. Savvides, C.-C. Han, and M. B. Srivastava, "Dynamic fine-grained localization in ad-hoc networks of sensors," in *Proc. ACM MobiCom 2001*, (Rome, Italy), July 2001, pp. 166–179.

[11] S. Capkun, M. Hamdi, and J. Hubaux, "GPS-free positioning in mobile ad-hoc networks," *Cluster Computing*, vol. 5, pp. 157–167, Apr. 2002.

[12] K. Langendoen and N. Reijers, "Distributed localization in wireless sensor networks: A quantitative comparison," *Computer Networks*, vol. 43, pp. 499–518, Nov. 2003.

[13] N. Bulusu, J. Heidimann, and D. Estrin, "GPS-less low-cost outdoor localization for very small devices," *IEEE Pers. Commun.*, vol. 7, pp. 28–34, Oct. 2000.

[14] D. Niculescu and B. Nath, "Ad hoc positioning system (APS)," in *Proc. IEEE GLOBECOM 2001*, (San Antonio, USA), Nov. 2001, pp. 2926–2931.

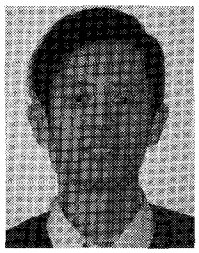
[15] C. Savarese, J. Rabaey, and K. Langendoen, "Robust positioning algorithms for distributed ad-hoc wireless sensor networks," in *Proc. USENIX Technical Annual Conference*, (Monterey, Canada), June 2002, pp. 317–327.

[16] T. He, C. Huang, B. M. Blum, J. A. Stankovic, and T. Abdelzaher, "Range-free localization schemes for large scale sensor networks," in *Proc. ACM MobiCom 2003*, (San Diego, USA), Sept. 2003, pp. 81–95.

[17] C.Liu and K. Wu, "Sensor localization with ring overlapping based on comparison of received signal strength indicator," in *Proc. IEEE MASS 2004*, (Fort Lauderdale, USA), Oct. 2004, pp. 516–518.

[18] N. B. Priyantha, H. Balakrishnan, E. Demaine, and S. Teller, "Anchor-free distributed localization in sensor networks," in *Proc. ACM SenSys 2003*, (Los Angeles, USA), Nov. 2003, pp. 340–341.

[19] A. Savvides, W. L. Garber, R. L. Moses, and M. B. Srivastava, "An analysis of error inducing parameters in multihop sensor node localization," *IEEE Trans. Mobile Computing*, vol. 4, pp. 567–577, Nov. 2005.



Yuan Zhang was born in September, 1974, in Shandong, China. He received the B.Sc. degree in Electronics Engineering from Nankai University, and the M.Sc. degree in Computer Engineering from Shandong University, China, in 1996 and 2003, respectively. From October 2003 to October 2004, he was a research fellow at Kyung Hee University, Korea. He is currently heading the Network Technology Institute and works as a lecturer at Jinan University, China. His research interests are in mobile communications and wireless networks, especially focusing on mobile ad hoc networks, sensor networks, and 3G cellular networks.

Wenwu Wu was born in May, 1970, in Sichuan, China. She received the B.Sc. degree in Computer Engineering from Xian University of Engineering Science and Technology, and M.Sc degree in Computer Engineering from Shandong University, in 1992 and 2002, respectively. She has been working as a network engineer and lecturer at the Network Information Center of Jinan University from 1997. Her research interests include network security, IPv6 networking, and wireless networks.

Yuehui Chen was born in 1964 in Shandong Province of China. He received his B.Sc. degree in mathematics/automatics from Shandong University, China, and Master and Ph.D. degrees in electrical engineering from the Kumamoto University of Japan in 1985, 1999 and 2001, respectively. During 2001–2003, he worked as the Senior Researcher at the Memory-Tech. Corporation, Tokyo. Since 2003, he has been a member at the faculty of School of Information Science and Engineering, Jinan University, where he is currently head of the Computational Intelligence Institute. His research interests include computational intelligence and wireless networks. He is the author and co-author of more than 70 technique papers. Professor Yuehui Chen is a member of IEEE, the IEEE Systems, Man and Cybernetics Society, and the Computational Intelligence Society. He is also a member of the editorial boards of several technical journals and a member of the program committee of several international conferences.